NASA Procedural Requirements

NPR 7150.2A
Effective Date: November 19, 2009
Expiration Date: November 19, 2014

COMPLIANCE IS MANDATORY

Printable Format (PDF)

Request Notification of Change (NASA Only)

Subject: NASA Software Engineering Requirements

Responsible Office: Office of the Chief Engineer

Chapter 2: Software Management Requirements

The software management activities define and control the many software aspects of a project from beginning to end. This includes the interfaces to other organizations, determination of deliverables, estimates and tracking of schedule and cost, risk management, formal and informal reviews as well as other forms of verification and validation, and determination of the amount of supporting services. The planned management of these activities is captured in one or more software and/or system plans.

2.1 Compliance with Laws, Policies, and Requirements

The software management process requires the understanding and application of laws and additional NASA policy requirements that impact the development, release, and/or maintenance of software. The documents listed in this section are additional requirements that may have an affect on software development projects and are mentioned here for awareness and completeness.

2.1.1 The project ensures that software invention requirements of NPD 2091.1, Inventions by Government Employees, are implemented by the project.

2.1.2 The project ensures that software technology transfer requirements of NPR 2190.1, NASA Export Control Program, are implemented by the project. The project ensures that there will be no access by foreign persons or export or transfer to foreign persons or destinations until an export control review is completed and access/release is approved in accordance with NPR 2190.1, NASA Export Control Program, and NPR 2210.1, External Release of Software.

2.1.3 The project ensures that software external release requirements of NPR 2210.1, External Release of NASA Software, are implemented by the project.

2.1.4 The project ensures that the information security requirements of NPD 2810.1, Security of Information Technology, are implemented by the project.

2.1.5 The project ensures that software is accessible to individuals with disabilities in accordance with 36 CFR Part 1194, Electronic and Information Technology Accessibility Standards.

2.1.6 The project ensures that software acquisitions or developments that meet NASA's capitalization criteria be capitalized per NASA's Property Plant and Equipment Policy, NPD 9250.1.

[Requirement numbers SWE-007 through SWE-012 are reserved.]

2.2 Software Life-Cycle Planning

Software Life-Cycle Planning covers the software aspects of a project from inception through retirement. Software Life-Cycle Planning cycle is an organizing process that considers the software as a whole and provides the planning activities required to insure a coordinated, well-engineered process for defining and implementing project activities. These processes, plans, and activities are coordinated within the project. At project conception, software needs for the project are analyzed, including acquisition, supply, development, operation, maintenance, retirement, and supporting activities and processes. The software effort is scoped and the processes, measurements, and activities are documented in software plan(s).

2.2.1 Software Plans

2.2.1.1 The project shall develop software plan(s). [SWE-013]

Note: The requirement for the content of each software plan (whether stand-alone or condensed into one or more project level or software documents) is defined in Chapter 5. These include, but are not limited to:

a. Software development or management plan.

b. Software configuration management plan.

c. Software test plans.

d. Software maintenance plans.

e. Software assurance plans.

Note: Software engineering and the software assurance disciplines are integrally related and yet each has its own responsibilities. Jointly they are responsible for providing project management with the optimal solution for software to meet the engineering, safety, quality, and reliability needs of the project. This necessitates a close working relationship to assure the appropriate levels of effort for both. See NASA-STD-8739.8 for more details on development of a software assurance plan.

2.2.1.2 For safety-critical software, the project shall develop a software safety plan. [SWE-130]

Note: The requirement for the content of the software safety plan (whether stand-alone or condensed into one or more project level or software documents) is defined in Chapter 5. The NASA Software Safety Standard, NASA-STD-8719.13, contains detailed requirements and guidance on development of software safety plans. Software engineering and the software safety disciplines jointly are responsible for providing project management with the optimal solution for software to meet the engineering, safety, quality, and reliability needs of the project.

2.2.1.3 If a project is selected for software Independent Verification and Validation (IV&V) by the NASA Chief, Safety and Mission Assurance, the NASA IV&V program shall develop an IV&V Project Execution Plan (IPEP). [SWE-131]

Note: The selection of projects and capabilities by the Chief, Safety and Mission Assurance, for IV&V is based on recommendations from the NASA IV&V Board of Advisors. The recommendations provided by the NASA IV&V Board of Advisors are the result of a multi-step process. First, the NASA Software Inventory, maintained by the Office of the Chief Engineer, is used to generate a candidate list of projects. The candidate project criteria include:

a. Whether the project contains safety-critical software.

b. The project's software classification(s) per this NPR.

c. The level of software effort on the project.

d. The project's NASA Category (e.g., Category 1, 2, or 3, as defined in NPR 7120.5D).

e. Whether the responsible Center recommends the project for IV&V.

Second, the IV&V program utilizes this list and associated Center-provided data, captured in the NASA Software Inventory, to initiate a portfolio based risk assessment (PBRA) for capabilities within the candidate projects. The PBRA process assesses the risk of capabilities which have software associated with them, based on factors including:

a. Complexity of the software implementing the capability under assessment.

b. Risk to safety and mission success associated with the capability under assessment.

c. Consequences of failure of the capability under assessment.

d. Time to criticality (i.e., the amount of time before the system or subsystem enters into a critical condition).

The results of these assessments are prioritized, by capability, in order to provide recommendations for IV&V support on specific capabilities associated with specific projects. These recommendations are provided to the NASA IV&V Board of Advisors, which in turn provides its recommendations to the Chief, Safety and Mission Assurance.

Note: The IV&V program determines and documents the services to be provided for projects selected for IV&V by the NASA Chief, Safety and Mission Assurance. IV&V support is funded and managed independent of the selected project. The IPEP is developed by the IV&V program and serves as the operational document that will be provided to the project receiving IV&V support. The IV&V program and the project will establish a mutual understanding of the NASA IV&V program's activities and IV&V project interfaces. Per the responsibilities defined in NPD 7120.4, NASA Engineering and Program/Project Management Policy, section 5.J.(5), projects ensure that software providers allow access to software and associated artifacts to enable implementation of IV&V. Additional information on the content of the IPEP is found in Chapter 5. Additional detail on the IV&V PBRA and IPEP may be found in the NASA IV&V Management System, located at: http://www.nasa.gov/centers/ivv/ims/home/index.html

2.2.2 The project shall implement, maintain, and execute the software plan(s). [SWE-014]

2.2.3 The project shall establish, document, and maintain at least one software cost estimate and associated cost parameter(s) that satisfies the following conditions: [SWE-015]

a. Covers the entire software life cycle.

b. Is based on selected project attributes (e.g., assessment of the size, functionality, complexity, criticality, and risk of the software processes and products).

c. Is based on the cost implications of the technology to be used and the required maturation of that technology.

Note: In the event there is a decision to outsource, it is a best practice that both the acquirer (NASA) and the provider (contractor/sub) should be responsible for developing software cost estimates.

2.2.4 The project shall document and maintain a software schedule that satisfies the following conditions: [SWE-016]

a. Coordinates with the overall project schedule.

b. Documents the interactions of milestones and deliverables between software, hardware, operations, and the rest of the system.

c. Reflects the critical path for the software development activities.

2.2.5 The project shall plan, track, and ensure project specific software training for project personnel. [SWE-017]

Note: This requirement is intended to address skills needed to support the software activities on a project. Not all skills required to accomplish a software project are "software" skills. The software engineering staff may require training in other domains to facilitate support of a project.

2.2.6 The project shall regularly hold reviews of software activities, status, and results with the project stakeholders and track issues to resolution. [SWE-018]

2.2.7 The project shall select and document a software development life cycle or model that includes phase transition criteria for each life-cycle phase (e.g., formal review milestones, informal reviews, software requirements review (SRR), preliminary design review (PDR), critical design review (CDR), test readiness reviews, customer acceptance or approval reviews). [SWE-019]

2.2.8 Software Assessments

2.2.8.1 The project shall classify each system and subsystem containing software in accordance with the software classification definitions for Classes A, B, C, D, E, F, G, and H software in Appendix E. [SWE-020]

Note: The applicability of requirements in this NPR to specific systems and subsystems containing software is determined through the use of the NASA-wide definitions for software classes in Appendix E and the designation of the software as safety-critical or non safety-critical in conjunction with the Requirements Mapping Matrix in Appendix D. These definitions are based on 1) usage of the software with or within a NASA system, 2) criticality of the system to NASA's major programs and projects, 3) extent to which humans depend upon the system, 4) developmental and operational complexity, and 5) extent of the Agency's investment. The NPR allows software written to support development activities (e.g., verification software, simulations) to be classified separately from the system under development; however, such support software is usually developed in conjunction with the system and not as a separate project. Projects may decide to reuse processes and work products established for the development of the system to satisfy the NPR 7150.2 requirements for the support software. The software assurance organization will perform an independent classification assessment and the results will be compared as per NASA-STD-8739.8, Software Assurance Standard. Software management and software assurance must reach agreement on classification of systems and subsystems. Disagreements are elevated via both the Engineering Technical Authority and Safety and Mission Assurance Technical Authority chains. The classification decision is documented in the Software Development or Management Plan as defined in Chapter 5.

2.2.8.2 The project's software assurance organization shall perform an independent classification assessment. [SWE-132]

2.2.8.3 The project, in conjunction with the Safety and Mission Assurance organization, shall determine the software safety criticality in accordance with NASA-STD-8739.8. [SWE-133]

Note: Software safety criticality is initially determined in the formulation phase using the NASA Software Assurance Standard, NASA-STD-8739.8. As the software is developed or changed and the computer software configuration items (CSCI), models, and simulations are identified, the safety-critical software determination can be reassessed and applied at lower levels. The software safety assessment and planning are performed for each software acquisition, development, and maintenance activity, and for changes to legacy\heritage systems. When software in a system or subsystem is found to be safety critical, additional requirements in the NASA Software Safety Standard will augment those associated with the software class requirements found in this document. The software assurance organization is required by NASA Software Assurance Standard, NASA-STD-8739.8, to perform an independent software safety criticality assessment and work with the project to resolve any differences. Engineering and software assurance must reach agreement on safety-critical determination of software. Disagreements are elevated via both the Engineering Technical Authority and Safety and Mission Assurance Technical Authority chains.

2.2.9 If a system or subsystem evolves to a higher software classification as defined in Appendix E, then the project shall update its plan to fulfill the added requirements per the Requirements Mapping Matrix in Appendix D. [SWE-021]

2.2.10 The project shall implement software assurance per NASA-STD-8739.8, NASA Software Assurance Standard. [SWE-022]

Note: Software assurance activities occur throughout the life of the project. Some of the actual analyses and activities may be performed by engineering or the project. NASA's Safety and Mission Assurance organizations provide assurance that the products and processes are implemented according to the agreed upon plan(s). Software assurance is recommended on software activities and products, including solicitations, contract and memorandums of agreements, software plans, requirements, design, implementation, verification, validation, certification, acceptance, maintenance, operations, and retirement activities.

2.2.11 When a project is determined to have safety-critical software, the project shall ensure that the safety requirements of NASA-STD-8719.13, Software Safety Standard, are implemented by the project. [SWE-023]

NOTE: Engineering and software assurance initially determine software safety criticality in the formulation phase per NASA-STD-8739.8, NASA Software Assurance Standard; the results are compared and any differences are resolved. As the software is developed or changed and the software components, software models, and software simulations are identified, the safety-critical software determination can be reassessed and applied at lower levels. Further scoping and tailoring of the safety effort is found in the NASA-STD-8719.13, Software Safety Standard and NASA-GB-8719.13, NASA Software Safety Guidebook.

2.2.12 When a project is determined to have safety-critical software, the project shall ensure the following items are implemented in the software: [SWE-134]

a. Safety-critical software is initialized, at first start and at restarts, to a known safe state.

b. Safety-critical software safely transitions between all predefined known states.

c. Termination performed by software of safety critical functions is performed to a known safe state.

d. Operator overrides of safety-critical software functions require at least two independent actions by an operator.

e. Safety-critical software rejects commands received out of sequence, when execution of those commands out of sequence can cause a hazard.

f. Safety-critical software detects inadvertent memory modification and recovers to a known safe state.

g. Safety-critical software performs integrity checks on inputs and outputs to/from the software system.

h. Safety-critical software performs prerequisite checks prior to the execution of safety-critical software commands.

i. No single software event or action is allowed to initiate an identified hazard.

j. Safety-critical software responds to an off nominal condition within the time needed to prevent a hazardous event.

k. Software provides error handling of safety-critical functions.

l. Safety-critical software has the capability to place the system into a safe state.

m. Safety-critical elements (requirements, design elements, code components, and interfaces) are uniquely identified as safety-critical. n. Incorporate requirements in the coding methods, standards, and/or criteria to clearly identify safety-critical code and data within source code comments.

Note: This section provides additional software safety requirements that are considered a best practice for safety-critical systems incorporating safety-critical software. These requirements are applicable to components that reside in a safety-critical system, and the components control, mitigate or contribute to a hazard as well as software used to command hazardous operations/activities. The requirements contained in this section complement the processes identified in NASA-STD-8719.13, NASA Software Safety Standard. Software engineering and software assurance disciplines each have specific responsibilities for providing project management with work products that meet the engineering, safety, quality, and reliability requirements on a project.

Note: Additional notes on specific items addressed above are:

Item a. - Aspects to consider when establishing a known safe state includes state of the hardware and software, operational phase, device capability, configuration, file allocation tables, and boot code in memory.

Item d. - Multiple independent actions by the operator help to reduce potential operator mistakes.

Item f. - Memory modifications may occur due to radiation-induced errors, uplink errors, configuration errors, or other causes so the computing system must be able to detect the problem and recover to a safe state. As an example, computing systems may implement error detection and correction, software executable and data load authentication, periodic memory scrub, and space partitioning to provide protection against inadvertent memory modification. Features of the processor and/or operating system can be utilized to protect against incorrect memory use.

Item g. - Software needs to accommodate both nominal inputs (within specifications) and off-nominal inputs, from which recovery may be required.

Item h. - The requirement is intended to preclude the inappropriate sequencing of commands. Appropriateness is determined by the project and conditions designed into the safety-critical system. Safety-critical software commands are commands that can cause or contribute to a hazardous event or operation.

Item j. - The intent is to establish a safe state following detection of an off-nominal indication. The safety mitigation must complete between the time that the off-nominal condition is detected and the time the hazard would occur without the mitigation. The safe state can either be an alternate state from normal operations or can be accomplished by detecting and correcting the fault or failure within the timeframe necessary to prevent a hazard and continuing with normal operations.

Item k.- Error handling is an implementation mechanism or design technique by which software faults and/or failures are detected, isolated, and recovered to allow for correct run-time program execution. The software error handling features that support safety-critical functions may detect and respond to hardware and operational faults and/or failures.

Item l.- The design of the system must provide sufficient sensors and effectors, as well as self checks within the software, in order to enable the software to detect and respond to system potential hazards.

2.2.13 The project shall ensure that actual results and performance of software activities are tracked against the software plans. [SWE-024]

2.2.14 The project shall ensure that corrective actions are taken, recorded, and managed to closure when actual results and performance deviate from the software plans. [SWE-025]

2.2.15 The project shall ensure that changes to commitments (e.g., software plans) are agreed to by the affected groups and individuals. [SWE-026]

2.3 Commercial, Government, Legacy\Heritage and Modified Off-The-Shelf Software

Projects utilizing Commercial, Government, Legacy\Heritage, and MOTS software components need to take into consideration the importance of planning and managing the inclusion of those components into the project software. The off-the-shelf software discussed here apply only when the off-the-shelf software elements are to be included as part of a NASA system (per section P.2.1). The following requirements do not apply to stand-alone desktop applications (e.g., word processing programs, spreadsheet programs, presentation programs). When software components use COTS applications (e.g., spreadsheet programs, database programs) within a NASA system/subsystem application, the software components need to be assessed and classified as part of the software subsystem in which they reside. Note that Commercial, Government, Legacy\Heritage, and MOTS software must also meet the applicable requirements for each class of software.

2.3.1 The project shall ensure that when a COTS, GOTS, MOTS, reused, or open source software component is to be acquired or used, the following conditions are satisfied: [SWE-027]

a. The requirements that are to be met by the software component are identified.

b. The software component includes documentation to fulfill its intended purpose (e.g., usage instructions).

c. Proprietary, usage, ownership, warranty, licensing rights, and transfer rights have been addressed.

d. Future support for the software product is planned.

e. The software component is verified and validated to the same level of confidence as would be required of the developed software component.

Note: The project responsible for procuring off-the-shelf software is responsible for documenting, prior to procurement, a plan for verifying and validating the off-the-shelf software to the same level of confidence that would be needed for an equivalent class of software if obtained through a "development" process. The project ensures that the COTS, GOTS, MOTS, reused, and open source software components and data meet the applicable requirements in this NPR assigned to its software classification as shown in Appendix D.

Note: For these types of software components consider the following:

a. Supplier agreement to deliver or escrow source code or third party maintenance agreement is in place.

b. A risk mitigation plan to cover the following cases is available:

(1) Loss of supplier or third party support for the product.

(2) Loss of maintenance for the product (or product version).

(3) Loss of the product (e.g., license revoked, recall of product, etc.).

c. Agreement that the project has access to defects discovered by the community of users has been obtained. When available, the project can consider joining a product users group to obtain this information.

d. A plan to provide adequate support is in place; the plan needs to include maintenance planning and the cost of maintenance.

e. Documentation changes to the software management, development, operations, or maintenance plans that are affected by the use or incorporation of COTS, GOTS, MOTS, reused, and legacy\heritage software.

f. Open source software licenses review by the Center Counsel.

2.4 Software Verification and Validation

Ensuring that the software products meet their requirements and intended usage, and that the products were built correctly is the purpose of verification and validation. Both software validation and software verification activities span the entire software life cycle and need to be planned. Formal and informal reviews, software peer reviews/inspections, testing, demonstration, and analyses can be used. Each project is generally free to choose the extent and combination of verification and validation methods and activities that best suit the project. Because software peer reviews/inspections are such an important verification and validation tool with proven value, specific software peer review/inspection requirements are contained in this NPR (Chapter 4).

2.4.1 The project shall plan software verification activities, methods, environments, and criteria for the project. [SWE-028]

Note: Software verification is a software engineering activity that shows confirmation that software products properly reflect the requirements specified for them. In other words, verification ensures that "you built it right." Examples of verification methods include but are not limited to: software peer reviews/inspections of software engineering products for discovery of defects, software verification of requirements by use of simulations, black box and white box testing techniques, software load testing, software stress testing, software performance testing, decision table-based testing, functional decomposition-based testing, acceptance testing, path coverage testing, analyses of requirement implementation, and software product demonstrations. Refer to the software plan requirements for software verification planning and incorporation, including the planned use of software IV&V activities.

2.4.2 The project shall plan the software validation activities, methods, environments, and criteria for the project. [SWE-029]

Note: Software validation is a software engineering activity that shows confirmation that the software product, as provided (or as it will be provided), fulfills its intended use in its intended environment. In other words, validation ensures that "you built the right thing." Examples of validation methods include but are not limited to: formal reviews, prototype demonstrations, functional demonstrations, software testing, software peer reviews/inspections of software product component, behavior in a simulated environment, acceptance testing against mathematical models, analyses, and operational environment demonstrations. Refer to the software plan requirements for software validation planning and incorporation (Chapter 5).

2.4.3 The project shall record, address, and track to closure the results of software verification activities. [SWE-030]

2.4.4 The project shall record, address, and track to closure the results of software validation activities. [SWE-031]

2.5 Project Formulation Requirements

Much of the project preparation and planning takes place during project formulation. Identification of project formulation requirements is an essential part of the early planning phases and must be started as the project begins. This is especially important as software requirements must be properly incorporated into the project cost estimates, schedule estimates, work planning, solicitations, evaluations of contractors, and the contracts themselves.

2.5.1 [SWE-032] The project shall ensure that software is acquired, developed, and maintained by an organization with a non-expired Capability Maturity Model Integration® for Development (CMMI-DEV) rating as measured by a Software Engineering Institute (SEI) authorized or certified lead appraiser as follows:

For **Class A** software:

CMMI-DEV Maturity Level 3 Rating or higher for software, or CMMI-DEV Capability Level 3 Rating or higher in all CMMI-DEV Maturity Level 2 and 3 process areas for software.

For **Class B** software:

CMMI-DEV Maturity Level 2 Rating or higher for software, or CMMI-DEV Capability Level 2 Rating or higher for software in the following process areas:

a. Requirements Management.

b. Configuration Management.

c. Process and Product Quality Assurance.

d. Measurement and Analysis.

e. Project Planning.

f. Project Monitoring and Control.

g. Supplier Agreement Management (if applicable).

For **Class C** software:

The required CMMI-DEV Maturity Level for Class C software will be defined per Center or project requirements.

Note: Organizations who have completed Standard CMMI® Appraisal Method for Process Improvement (SCAMPISM) Class A appraisals against the CMMI-DEV model are to maintain their rating and have their results posted on the SEI Web site so that NASA can assess the current maturity/capability rating. Software development organizations need to be reappraised and keep an active appraisal rating posted on the SEI Web site during the time that they are responsible for the development and maintenance of the software.

Note: For Class A software development only, a transition period to obtain a CMMI-DEV Maturity/Capability Level 3 Rating will be allowed for organizations developing Class A software per the NASA Headquarters' Office of the Chief Engineer's approved Center Software Engineering Improvement Plan as described in SWE-003, SWE-004, and SWE-108.

Note: For Class B software, in lieu of a CMMI rating by a development organization, the project will conduct an evaluation, performed by a qualified evaluator selected by the Center Engineering Technical Authority, of the seven process areas listed in SWE-032 and mitigate any risk, if deficient. This exception is intended to be used in those cases in which NASA wishes to purchase a product from the "best of class provider," but the best of class provider does not have the required CMMI rating. When this exception is exercised, the Center Engineering Technical Authority should be notified.

2.5.2 The project shall assess options for software acquisition versus development. [SWE-033]

Note: The assessment can include risk, cost, and benefits criteria for each of the options listed below:

a. Acquire an off-the-shelf software product that satisfies the requirement.

b. Develop the software product or obtain the software service internally.

c. Develop the software product or obtain the software service through contract.

d. Enhance an existing software product or service.

Note: Risks are considered in software make/buy and acquisition decisions. The project needs to ensure that software products used in the design or support of human space flight components or systems include a level of rigor in risk mitigation as a software management requirement, regardless of software classification. The level of documentation needed for risk identification and tracking is defined by the Center processes.

2.5.3 The project shall define and document or record the acceptance criteria and conditions for the software. [SWE-034]

2.5.4 For new contracts, the project shall establish a procedure for software supplier selection, including proposal evaluation criteria. [SWE-035]

2.5.5 The project shall determine which software processes, activities, and tasks are required for the project. [SWE-036]

2.5.6 The project shall define the milestones at which the software supplier(s) progress will be reviewed and audited as a part of the acquisition activities. [SWE-037]

Note: Known contract milestones are expected to be included in the resulting contract.

2.5.7 The project shall document software acquisition planning decisions. [SWE-038]

Note: This may be in an acquisition plan or in another project planning document.

2.6 Software Contract Requirements

The requirements in this section are applicable for NASA contracted software procurements (e.g., reuse of existing software, modification of existing software, contracted and subcontracted software, and/or development of new software). Acquisition requirements are focused both inside the acquisition organization to ensure the acquisition is conducted effectively and outside the acquisition organization as the organization conducts project monitoring and control of its suppliers. These acquisition requirements provide a foundation for acquisition process discipline and rigor that enables product and service development to be repeatedly executed with high levels of acquisition success. This section contains project software acquisition and contract requirements to ensure that NASA has the data needed for the review of project provided systems and/or services. The project is responsible for ensuring that these requirements apply when software activities are subcontracted from a NASA prime contractor. These requirements are used in addition to, not in place of, the other requirements of this NPR.

2.6.1 Government software insight requirements.

2.6.1.1 The project shall require the software supplier(s) to provide insight into software development and test activities; at a minimum the following activities are required: monitoring integration, review of the verification adequacy, review of trade study data and results, auditing the software development process, participation in software reviews and systems and software technical interchange meetings. [SWE-039]

2.6.1.2 The project shall require the software supplier(s) to provide NASA with all software products and software process tracking information, in electronic format, including software development and management metrics. [SWE-040]

2.6.1.3 The project shall require the software supplier(s) to notify the project, in the response to the solicitation, as to whether open source software will be included in code developed for the project. [SWE-041]

2.6.1.4 The project shall require the software supplier(s) to provide NASA with electronic access to the source code developed for the project, including MOTS software and non-flight software (e.g., ground test software, simulations, ground analysis software, ground control software, science data processing software, and hardware manufacturing software). [SWE-042]

Note: Known contract requirements are addressed in the solicitations and included in the resulting contract. Additionally, if the project needs to control further use and distribution of the resulting software or requires unlimited rights in the software, e.g., right to use, modify, and distribute the software for any purpose, the project can consider having the software copyright assigned to the Government. A list of software deliverables needs to be addressed in the solicitations, if the software is being procured. The project can consult with the Center's Chief of Patent/Intellectual Property Counsel regarding required rights associated with the software. Section 3.5.4 contains a list of Software Operations, Maintenance, and Retirement deliverables.

2.6.2 Supplier Monitoring Requirements.

2.6.2.1 The project shall require the software supplier to track all software changes and non-conformances and provide the data for the project's review. [SWE-043]

2.6.2.2 The project shall require the software supplier(s) to provide software metric data as defined in the project's Software Metrics Report. [SWE-044]

Note: The requirement for the content of a Software Metrics Report is defined in Chapter 5.

2.6.2.3 The project shall participate in any joint NASA/contractor audits of the software development process and software configuration management process. [SWE-045]

2.6.2.4 The project shall require the software supplier(s) to provide a software schedule for the project's review and schedule updates as requested. [SWE-046]

2.6.2.5 The project shall require the software supplier(s) to make available, electronically, the software traceability data for the project's review. [SWE-047]

2.6.2.6 The project shall document in the solicitation the software processes, activities, and tasks to be performed by the supplier. [SWE-048]

| TOC | Preface | Chapter1 | Chapter2 | Chapter3 | Chapter4 | Chapter5 | Chapter6 | AppendixA | AppendixB | AppendixC | AppendixD | AppendixE | ALL |

| NODIS Library | Program Formulation(7000s) | Search |

DISTRIBUTION:
NODIS